

UNIVERSITÉ DE MONTRÉAL

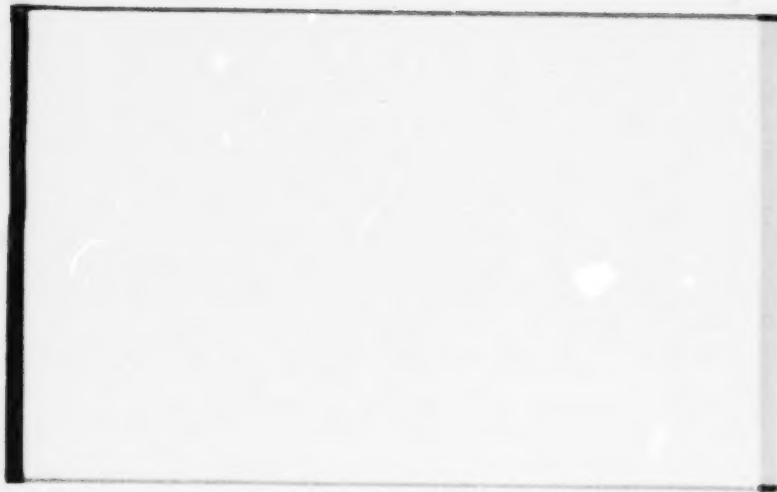


CENTRE DE RECHERCHE SUR LES TRANSPORTS



CASE POSTALE 6128, STATION A, MONTRÉAL, QUÉBEC, H3C 3J7

88-06301/



A MATHEMATICAL PROGRAMMING APPROACH TO
EQUILIBRIUM METHODS OF TRAFFIC ASSIGNMENT
WITH FIXED DEMANDS

BY

Sang Nguyen

This research was supported by a grant from the Ford Motor Company of Canada for applied research in the Analysis and Control of Urban Transportation Networks.

Université de Montréal
Publication # 138 - Département d'informatique
Publication # 18 - Centre de recherche sur les transports
Août 1973

Dépôt légal - 1er trimestre 1978

ABSTRACT

This paper is concerned with the traffic assignment problem with fixed origin-destination demands and increasing travelling cost functions.

Our purpose is to present a unified approach to equilibrium methods for traffic assignment, that not only provides a class of iterative algorithms but also gives a new insight into some existing methods.

All the developed algorithms were coded in Fortran for the CDC CYBER 74. Coding considerations and computational efficiencies are reported.

RESUME

Nous nous proposons d'étudier le problème de l'affectation du trafic routier dans lequel le coût de parcours d'un tronçon de route est une fonction croissante de sa charge, et où le nombre de véhicules allant d'une zone spécifique à une autre est supposé constant.

Notre but est de présenter une approche unifiée des méthodes d'équilibre de résolution, qui tout en permettant de produire une classe d'algorithmes opérationnels, apporte une meilleure compréhension de certaines méthodes existantes.

Tous les algorithmes développés ont été programmés en Fortran. Le codage et l'essai des programmes sur l'ordinateur CDC CYBER 74 sont rapportés.



1. INTRODUCTION

Traffic assignment is one of the key problems encountered in the evaluation of development plans in a transportation network.

This paper is concerned mainly with the assignment of vehicular traffic in an urban network. After the urban network has been schematized in its main arterials, the city divided into a number of zones and the flow of vehicles from one zone to another determined, the assignment problem is the determination of the itineraries taken by drivers and the resulting load on the network links.

The driver's route choice depends on a large number of factors; for example, the travelling time, the route length and its condition, the number of left turns encountered, etc... For this reason, it is a very complex task to simulate with precision the driver's behaviour. However, if certain behavioural hypotheses on the driver's route choice are made and in addition the flow demands between zones are assumed to be constant with respect to time, it will be then possible to build a static model of the traffic assignment problem. The main hypotheses are:

- A. the perceived travelling cost of a road section is an increasing function of its load, and the cost of an itinerary is equal to the sum of the costs of its sections (links)
- B. the driver has complete information on the travelling cost of a specific link, and chooses his route accordingly so as to minimize the total cost of his trip.

The result of these hypotheses is that

at equilibrium all routes chosen from any origin to any destination will have equal travel costs, and that unused routes will have higher travel cost (Wardrop's First Principle [1]).

It was shown by Jorgensen [2], Jewell [3] and Dafermos [4], among others, that it is possible to formulate the above equilibrium model as a min-convex-cost multicommodity flow problem.

This latter formulation is most useful in the analysis of a transportation network, where the traffic assignment has to be done repeatedly by allowing the exploitation of known postoptimization techniques of mathematical programming.

The objective of this work is to investigate a class of iterative solution algorithms within a well known mathematical programming framework: the feasible directions method. A preliminary study of the relative efficiencies of these algorithms is reported. It appears, that some of the known algorithms (e.g. Bruynooghe, Gibert and Sakarovitch [5], Dafermos and Sparrow [6]), which are seemingly different in approach, fit well into this framework.

The material presented is listed in the following order: the mathematical statement of the problem is presented in the second section. The third section describes the mathematical programming framework of feasible direction methods and presents different specific descent directions. Several corresponding algorithms are developed in the fourth section. Section five covers numerical examples, discussions on the coding aspects of the algorithms and interpretations of computational results on test problems.

2. MATHEMATICAL STATEMENT OF THE PROBLEM

The transportation network is usually represented by a directed network, formed by a finite number of nodes $x \in N$ and links $(x,y) \in M$. A node describes either an intersection of roads or a zone, usually referred as a centroid. A road section with two-way traffic is represented by two links of opposite directions. A path is then a sequence of links, all oriented in the same direction, and free of loops. Associated to each pair of origin-destination is a fixed demand, and to each arc is a non-negative increasing travelling cost function.

The multicommodity flow model of the traffic assignment implies the distinction of the flow of vehicles coming out of each origin zone in the node-arc formulation, and the flow of vehicles related to each pair of origin-destination (O/D pair) in the arc-chain equivalent formulation.

The following notation will be used:

N	the set of nodes
n	the total number of nodes
M	the set of links
m	the total number of links
S	the set of origins
ns	the total number of origins
T	the set of destinations
(x,y)	a typical arc

- s a typical origin
 t a typical destination
 (s,t) a typical O/D pair
 np the total number of pairs of O/D
 Q_{st} the set of paths connecting origin s to destination t ;
 and sometimes simply Q_k , where k stands for the k^{th}
 O/D pair
 q a typical path
 f_{xy}^s the flow on the link (x,y) coming from the origin s
 h_q the flow on path q
 f_{xy} the total flow on link (x,y)
 g_{st} the flow demand of the pair (s,t) (or g_k)
 $c_{xy}(f_{xy})$ the travel cost of link (x,y) ; assumed monotone
 increasing

The optimization problem is then:

minimize

$$(2.1) \quad Z = \sum_{xy} z_{xy}(f_{xy})$$

subject to

$$(2.2) \quad \sum_{y \in N} f_{xy}^s - \sum_{y \in N} f_{yx}^s = \begin{cases} \sum_t g_{st} & , \quad x = s \\ -g_{st} & , \quad x = t \\ 0 & \text{otherwise} \end{cases}$$

for $s \in S$

$$(2.3) \quad f_{xy}^s > 0 \quad \text{for all } s, \text{ all } (x,y)$$

$$(2.4) \quad f_{xy} = \sum_s f_{xy}^s$$

where

$$(2.5) \quad g_{xy}(f_{xy}) = \int_0^{f_{xy}} c_{xy}(\sigma) d\sigma.$$

The equivalent arc-chain formulation is:

minimize

$$(2.6) \quad g = \sum_{xy} g_{xy}(f_{xy})$$

$$(2.7) \quad \sum_{q \in Q_k} h_q = g_k \quad k = 1, \dots, np$$

$$(2.8) \quad h_q > 0 \quad q \in Q = \bigcup_k Q_k$$

$$(2.9) \quad f_{xy} = \sum_q a_{xy}^q h_q$$

where a_{xy}^q equals 1 if $(x,y) \in q$ and 0 otherwise.

3. A UNIFIED APPROACH TO EQUILIBRIUM METHODS

Specially relevant to this study are primal methods in nonlinear programming. In particular, methods that are closely related to the well known feasible directions method of Zoutendijk [8]. The principle of this method entails that, given an initial feasible solution of a nonlinear program, a sequence of feasible solutions is generated. Each of these improves the objective function and is obtained from the previous one by determining

- (A) a direction that leads to an improved solution
- (B) and the optimal step length to be taken in this direction.

Before outlining the descent directions that will be used, we first give a formal statement of the general framework of the feasible directions method, as applied to a linearly constrained convex program.

Consider the problem

$$\begin{aligned}
 (3.1) \quad & \min Z(X) \\
 & \text{subject to} \quad AX - b = 0 \\
 & \quad \quad \quad X \geq 0,
 \end{aligned}$$

where $Z(X)$ is convex and continuously differentiable and the dimension of matrix A is $m \times n$.

We shall also use the following notation:

$$\nabla Z^t(X) = \left(\frac{\partial Z}{\partial x_1}, \dots, \frac{\partial Z}{\partial x_n} \right)$$

$$E = \{ X \mid AX - b = 0, X \geq 0 \}$$

$$K(X) = \{ i \mid x_i = 0, X \in E \}$$

$$D(X) = \{ d \mid Ad = 0, d_i \geq 0 \text{ if } i \in K(X) \}$$

X^l $X \in E$ at iteration l .

A vector d is a descent direction if it verifies

$$(3.2) \quad d \in D(X), \quad X \in E$$

and

$$(3.3) \quad \nabla g^t(X) d < 0.$$

Relation (3.2) implies that for any scalar λ , the point $X' = X + \lambda d$ verifies $AX' - b = 0$, while relation (3.3) implies that there exists a scalar $\sigma > 0$ such that $g(X + \lambda d) < g(X)$, $0 < \lambda \leq \sigma$.

A description of the general framework for feasible direction algorithms is as follows:

step 1 Let X^0 be an initial feasible direction

Set $l = 0$ and go to step 2.

step 2 If X^l verifies some stopping criterion that determines the optimality or near optimality of the solution, terminate otherwise go to step 3.

step 3 Find a direction d^l verifying $d^l \in D(X^l)$ and $\nabla g^t(X^l) d^l < 0$. Go to step 4.

step 4 Solve the one dimensional optimization problem

$$\min \{z(x^l + \lambda d^l) \mid \lambda \in L\}$$

where $L = \{\lambda \mid x^l + \lambda d^l \geq 0\}$. Let λ^l be the solution. Go to step 5.

step 5 Compute $x^{l+1} = x^l + \lambda^l d^l$.

Set $l = l + 1$ and go to step 2.

3.1 A DESCENT DIRECTION OBTAINED BY LINEAR APPROXIMATION

This direction is deduced from an algorithm suggested by Frank and Wolfe [9].

The linear approximation $z_a(\cdot)$ to the objective function $z(X)$ at x^l is

$$(3.4) \quad z_a(Y) = z(x^l) + \nabla z^t(x^l)(Y - x^l).$$

This suggests the following descent direction (referred later as the linear approximation direction or LAD) :

$$(3.5) \quad d^l = (Y^* - x^l)$$

where Y^* is an optimal solution of the linear program:

$$(3.6) \quad \begin{array}{ll} \text{minimize} & F = \nabla z^t(x^l) Y \\ \text{subject to} & \end{array}$$

$$\begin{array}{l} AY - b = 0 \\ Y \geq 0. \end{array}$$

It is clear that

$$d^l = Y^* - x^l \in D(x^l)$$

and

$$(3.7) \quad \nabla z^t(x^l) Y^* - \nabla z^t(x^l) x^l \leq 0,$$

where equality in relation (3.7) is verified if X^l is an optimal solution of problem (3.1).

3.2 DIRECTIONS OBTAINED BY PARTITIONING THE VARIABLES INTO BASIC AND NON-BASIC VARIABLES

Let B be a basis of A , X_I and X_J the basic and non-basic components of $X \in E$. In an obvious notation we may write

$$Ad = Bd_I + Cd_J,$$

$$\nabla Z^t(X)d = \nabla Z^t(X_I) d_I + \nabla Z^t(X_J) d_J.$$

Taking into consideration this partition of variables in equation

(3.2) and (3.3) we may obtain

$$(3.8) \quad d_I = -(B^{-1}C) d_J,$$

$$(3.9) \quad [\nabla Z(X_J) - (B^{-1}C)^t \nabla Z(X_I)]^t d_J < 0,$$

$$(3.10) \quad d_j > 0 \quad \text{if} \quad x_j = 0 \quad j \in J,$$

$$[(B^{-1}C) d_J]_i \leq 0 \quad \text{if} \quad x_i = 0 \quad i \in I.$$

We will now describe two descent directions that used the above partitioning scheme. The first

was suggested by Wolfe [10] in the Reduced Gradient method and the second, which is a particular case of the first one, was suggested by Zangwill [11] in the Convex Simplex method.

The Reduced Gradient direction (RGD) is defined as:

$$(3.11) \quad d_j = 0 \quad \text{if} \quad u_j > 0 \quad \text{and} \quad x_j = 0, \quad j \in J$$

$$d_j = -u_j \quad \text{otherwise,} \quad j \in J$$

$$d_I = -(B^{-1}C) d_J$$

where

$$(3.12) \quad u_j = \frac{\partial Z}{\partial x_j} - (B^{-1}C) \nabla Z(x_I) .$$

Let

$$u_{s1} = \min \{ u_j \}$$

and

$$u_{s2} x_{s2} = \max \{ u_j x_j \} ,$$

then the Convex Simplex direction (CSD) is defined as:

$$(3.13) \quad \text{If} \quad u_{s1} = u_{s2} x_{s2} = 0, \quad d = 0 .$$

$$\text{If} \quad |u_{s1}| > u_{s2} x_{s2}$$

$$\text{then} \quad d_j = 0 \quad \text{if} \quad j \neq s1, \quad j \in J$$

$$d_{s1} = 1$$

$$d_I = -(B^{-1}C) d_J ;$$

$$\text{otherwise} \quad d_j = 0 \quad \text{if} \quad j \neq s2, \quad j \in J$$

$$d_{s2} = -1$$

$$d_I = -(B^{-1}C) d_J .$$

The convergence of the feasible directions method will be insured if the descent direction generated in step 3 is closed (see for instance Zangwill [11]). Among the described directions the RGD is not closed, however it can be trivially modified to overcome the risk of jamming (for further details, see Luenberger [12]).

3.3 THE RESTRICTION STRATEGY

The arc-chain formulation requires the generation of all the possible paths connecting all the O/D pairs. This is not an easy task, when one knows that the number of paths grows exponentially with the number of nodes of the transportation network. As expected, often a fairly small proportion of these paths have positive flow at equilibrium. This strongly suggests, that a solution strategy which reduces the optimization problem to a related sequence of small subproblems, where each dealing explicitly with a very restricted set of variables, will prove to be useful.

In fact, such technique exists in mathematical programming and is referred by Geoffrion [13] as the "Restriction" strategy. An outline of the method will be given below without its theoretical justifications, which may be found in Zangwill [11] and Geoffrion [15].

Step 1 Let x^1 be a feasible solution of problem (3.1).

Set $K^1 = \{ i \mid x_i^1 = 0 \} ; \ell = 1 .$

Go to step 2.

step 2 Solve the subproblem

$$\begin{aligned}
 (3.15) \quad & \min z(Y) \\
 & \text{subject to} \quad AY - b = 0 \\
 & \quad Y \geq 0 \\
 & \quad y_i = 0 \quad \text{if } i \in K^l.
 \end{aligned}$$

Let Y^* and u^* be the optimal solution and associated multipliers of problem (3.15). The pair (Y^*, u^*) satisfies the Kuhn-Tucker optimality conditions for (3.15). Go to step 3.

Step 3 If all the $u_i^* > 0$ then terminate (Y^* is optimal for the original problem (3.1)); otherwise let $u_j^* = \min \{u_i^*\}$, set $K^{l+1} = K^l \cup \{j\}$, $l = l + 1$ and return to step 2.

It will be seen later that the application of the above strategy in conjunction with the outlined solution method, will allow the development of algorithms based on the arc-chain formulation which do not need the a priori generation of all paths.

4- SOLUTION ALGORITHMS OF THE EQUILIBRIUM MODEL

We are now ready to apply the mathematical programming approach, outlined in the previous section, to the traffic assignment problem (2.5) - (2.9).

4.1 ALGORITHM A1: AN APPLICATION OF THE LINEAR APPROXIMATION DIRECTION

The arc-chain formulation is used here, however it will be seen that the resulting algorithm does not depend on this formulation.

At iteration ℓ , the determination of the descent direction d^ℓ requires the solution of the following linear program:

$$(4.1) \quad \min \quad \nabla g^t Y = \sum_{q \in Q} \mu_q^\ell y_q$$

subject to

$$\sum_{q \in Q_k} y_q = g_k \quad k = 1, 2, \dots, np$$

where

$$(4.2) \quad \mu_q^\ell = \frac{\delta \sum_{xy} z_{xy} (f_{xy})}{\delta h_q^\ell} = \sum_{xy} a_{xy}^q c_{xy}^\ell$$

As μ_q^ℓ may be taken to be the length of the path q , the length of the arc (x,y) being c_{xy}^ℓ , the linear program (4.1) becomes merely a shortest path problem, and may be solved efficiently by any shortest path algorithm. Therefore, an optimal solution Y^* of problem (4.1) is given by

$$(4.4) \quad \begin{aligned} y_{p_k}^* &= g_k \quad \text{for } k = 1, 2, \dots, np \\ y_q^* &= 0 \quad \text{for } q \neq p_k \text{ and } q \in Q \end{aligned}$$

where p_k is the shortest path connecting the k^{th} O/D pair.

In this case, we may write for the descent direction d^ℓ :

$$\begin{aligned}
 (4.5) \quad d_{p_k}^\ell &= g_k - h_{p_k}^\ell & \text{for } k = 1, \dots, np \\
 d_q^\ell &= -h_q^\ell & \text{for } q \neq p_k \text{ and } q \in Q.
 \end{aligned}$$

On the other hand, the optimal step length λ^ℓ is given by the solution of the one dimensional optimization problem:

$$\begin{aligned}
 (4.6) \quad \min \quad & \sum_{xy} z_{xy} \left(\sum_q a_{xy}^q (h_q^\ell + \lambda d_q^\ell) \right) \\
 & 0 < \lambda \leq \min \{ -h_q^\ell / d_q^\ell \mid d_q^\ell < 0 \} = 1.
 \end{aligned}$$

Using (4.5) together with (4.6) we obtain

$$\begin{aligned}
 (4.7) \quad \min \quad & \sum_{xy} z_{xy} \left(f_{xy}^\ell + \lambda (w_{xy}^\ell - f_{xy}^\ell) \right) \\
 & 0 < \lambda \leq 1,
 \end{aligned}$$

where

$$\begin{aligned}
 (4.8) \quad w_{xy}^\ell &= \sum_k a_{xy}^{p_k} g_k \\
 f_{xy}^\ell &= \sum_q a_{xy}^q h_q^\ell.
 \end{aligned}$$

If the current flows $\{h_q^\ell\}$ are optimal, then equality $\nabla Z^t d^\ell = 0$ must be satisfied, that is

$$(4.9) \quad \sum_{xy} (w_{xy}^\ell - f_{xy}^\ell) c_{xy}^\ell = 0.$$

If it is not the case, the flows f_{xy}^ℓ will be then updated, using equations

$$(4.10) \quad f_{xy}^{\ell+1} = f_{xy}^\ell + \lambda^\ell (w_{xy}^\ell - f_{xy}^\ell).$$

Although the arc-chain formulation is used, we remark that knowledge of all the paths q is not necessary, since the variables

h_q^l are not involved in any of the computational steps just outlined. This is a very surprising and quite attractive result, because only the variables f_{xy} are needed to be kept along the solving process, therefore, reducing the number of variables of a multicommodity flow problem to the one of a single commodity flow problem.

The resulting algorithm is now stated.

ALGORITHM A1

Step 1 Let $\{f_{xy}^1\}$ be an initial feasible flow and $c_{xy}^1 = c_{xy}(f_{xy}^1)$ the associated link marginal costs. Set $l = 1$, go to step 2.

Step 2 Initialize $w_{xy}^l = 0$, $(x,y) \in M$.

For each pair of origin-destination k , find a shortest path P_k , and revise $w_{xy}^l = w_{xy}^l + g_k$ for all $(x,y) \in P_k$. Go to step 3.

Step 3 Set $d_{xy}^l = w_{xy}^l - f_{xy}^l$ for $(x,y) \in M$.

If $|\sum_{xy} c_{xy}^l d_{xy}^l| \leq \epsilon$ (positive scalar arbitrarily chosen) terminate, otherwise go to step 4.

Step 4 Compute λ^l minimizing $\sum_{xy} z_{xy} (f_{xy}^l + \lambda d_{xy}^l)$

subject to $0 < \lambda \leq 1$.

Go to step 5.

Step 5 Compute for all $(x,y) \in M$

$$f_{xy}^{l+1} = f_{xy}^l + \lambda^l d_{xy}^l$$

$$c_{xy}^{l+1} = c_{xy}(f_{xy}^{l+1}).$$

Set $\ell = \ell + 1$ and return to step 2.

Remarks

(A) Let $\{f_{xy}^*\}$ be the optimal flow and $\{f_{xy}^\ell\}$ the terminal solution given by the above algorithm, then by the convexity of the objective

function $Z = \sum_{xy} Z_{xy}(f_{xy})$ we obtain

$$Z^\ell - Z^* < - \nabla Z^\ell (f^* - f^\ell) ,$$

or

$$< - \nabla Z^\ell (w^\ell - f^\ell) ,$$

and finally $Z^\ell - Z^* < \epsilon$.

(B) If the link marginal cost is linear, that is

$$(4.11) \quad c_{xy}(f_{xy}) = \alpha_{xy} + \beta_{xy} f_{xy}$$

where

$$\alpha_{xy} > 0, \quad \beta_{xy} > 0,$$

then one verifies easily that the one dimensional optimization problem of step 4 is reduced to

$$(4.12) \quad \lambda^\ell = \min \left\{ 1, - \frac{\sum_{xy} c_{xy}^\ell (w_{xy}^\ell - f_{xy}^\ell)}{\sum_{xy} \beta_{xy} (w_{xy}^\ell - f_{xy}^\ell)^2} \right\} .$$

4.2

ALGORITHMS BASED ON THE ARC-CHAIN FORMULATION

The multicommodity flow optimization model which has been adopted for the traffic assignment, does not have any explicit capacity constraint; the interaction between the commodities is entirely reflected in the objective function. This particular structure allows to solve a sequence of single commodity problems instead of the original multi-

commodity problem. To this end the following solution strategy may be used:

- Step 1 Given an initial feasible flow of the original problem, set $i = 0$, $k = 0$ and go to step 2.
- Step 2 If $k = nc$ (total number of commodities) terminate, otherwise set $j = i \text{ modulo } (nc) + 1$ and go to step 3.
- Step 3 If the current flow is optimal for the j^{th} single commodity problem, set $k = k + 1$, $i = i + 1$ and go to step 2; otherwise solve the j^{th} subproblem, update the current flow, set $k = 1$, $i = i + 1$ and go to step 2.

Consider again the arc-chain formulation, with this formulation a commodity flow is characterized by a specific pair of origin - destination.

Consider the j^{th} origin-destination pair. The associated single commodity problem is:

$$(4.13) \quad \min z_j = \sum_{xy} z_{xy} (f_{xy})$$

subject to

$$\sum_{q \in Q_j} h_q = z_j$$

$$h_q \geq 0 \quad q \in Q_j$$

$$(4.14) \quad f_{xy} = \sum_{q \in Q_j} a_{xy}^q h_q + (\text{constant})$$

where the constant part of (4.14) is equal to $\sum_{k \neq j} \sum_{q \in Q_k} a_{xy}^q h_q$.

Let h_p^ℓ be the basic variable of problem 4.13 at iteration ℓ , and moreover suppose it is chosen accordingly to

$$(4.15) \quad \mu_p^\ell = \min \{ \mu_q^\ell, q \in Q_j \}$$

where
$$\mu_q^\ell = \sum_{xy} a_{xy}^q c_{xy}^\ell.$$

We may then write the descent direction (3.11) as:

$$(4.16) \quad \text{for } q \in \bar{Q}_j : \quad d_q^\ell = -\mu_q^\ell \text{ if } h_q^\ell > 0 \\ = 0 \text{ otherwise,}$$

and
$$d_p^\ell = - \sum_{q \in \bar{Q}} d_q^\ell$$

where
$$\mu_q^\ell = \mu_q^\ell - \mu_p^\ell$$

$$\bar{Q}_j = Q_j - \{p\}.$$

It should be noted that d_p^ℓ is strictly positive. Similarly, the CSD may be given by

$$(4.17) \quad d_q^\ell = 0 \quad q \in \bar{Q}_j \text{ and } q \neq r$$

$$d_r^\ell = -1$$

$$d_p^\ell = 1$$

where

$$(4.18) \quad u_r^\ell h_r^\ell = \max \{ u_q^\ell h_q^\ell \}.$$

It should be noted that $\min \{ u_q \} = 0$.

The choice criterion (4.15) of the basic variable is not unique, others may be just as valid. However, the choice of the direction (4.16) is suggested by the following situation: suppose one has an equilibrium flow over a set of paths connecting a specific O/D

pair, to which must be added a new path with a lesser travel cost. It seems natural, that to reestablish equilibrium over the new set of paths, one transfer a proportion of flow of each of the used path on the new path, this corresponds exactly to the effect produced by the descent direction (4.16)

Because of the similarity between the RGD and the CSD, we will only describe algorithms which use the RGD. Only minor modifications will be needed if algorithms corresponding to the CSD are desired.

In the case where all the possible paths connecting all the O/D pairs are available and stored, the algorithm resulting from the application of the general mathematical programming approach using the RGD will be as follows:

ALGORITHM A2

- Step 1 Given an initial flow $\{h_q\}$, $q \in Q_j$, $j = 1, \dots, np$,
set $i = -1$, $k = 0$ and go to step 2.
- Step 2 If $k = np$ terminate, otherwise set $i = i + 1$.
Set $j = i \text{ modulo } (np) + 1$, $\ell = 1$ and go to step 3.
- Step 3 Compute $c_{xy}^\ell = c_{xy}(f_{xy}^\ell)$, $(x, y) \in M$
and $\mu_q^\ell = \sum_{xy} a_{xy}^q c_{xy}^\ell$, $q \in Q_j$.
Determine $\mu_p = \min \{ \mu_q^\ell \}$
For $q \in \bar{Q}_j = Q_j - \{p\}$:
set $d_q^\ell = \mu_p^\ell - \mu_q^\ell$ if $h_q^\ell > \epsilon_1$ and $(\mu_q^\ell - \mu_p^\ell) / \mu_q^\ell > \epsilon_2$
= 0 otherwise.

$$\text{set } d_p^l = - \sum_{q \in Q_j} d_q^l$$

If $d_p^l = 0$ and $l = 1$, set $k = k + 1$ and go to step 2.

If $d_p^l = 0$ and $l > 1$, set $k = 1$ and go to step 2,

otherwise go to step 4.

Step 4 Solve for λ^l the following problem:

$$\min \sum_{xy} z_{xy} (f_{xy}^l + \lambda w_{xy}^l)$$

subject to

$$0 < \lambda \leq \min \{ -h_q^l / d_q^l \mid d_q^l < 0 \}$$

where

$$w_{xy}^l = \sum_{q \in Q_j} a_{xy}^q d_q^l.$$

Step 5 Reallocate the flow by setting

$$h_q^{l+1} = h_q^l + \lambda^l d_q^l, \quad q \in Q_j$$

$$f_{xy}^{l+1} = f_{xy}^l + \lambda^l w_{xy}^l, \quad (x,y) \in M.$$

Set $l = l + 1$ and go to step 3.

Remarks

(A) Let $\{h_q^*\}$ be the optimal flow and $\{h_q^l\}$ be the terminal flow given by the above algorithm. It can be shown that

$$(4.19) \quad z^l - z^* \leq \sum_{j=1}^{np} \left[\epsilon_1 \sum_{q \in Q_j^1} (\mu_q^l - \mu_p^l) + \epsilon_2 \sum_{q \in Q_j^2} \mu_q^l h_q^l \right]$$

where

$$h_q^l \leq \epsilon_1 \quad \text{for } q \in Q_j^1$$

and

$$h_q^\ell > \varepsilon_1 \quad \text{and} \quad (\mu_q^\ell - \mu_p^\ell) / \mu_q^\ell \leq \varepsilon_2 \quad \text{for } q \in Q_j^2 .$$

(B) In the case where $c_{xy} (f_{xy})$ is linear and given by (4.11), step 4 is reduced to

compute
$$\lambda^\ell = \min \{\lambda_1, \lambda_2\}$$

where

$$\lambda_1 = - \frac{\sum_{xy} c_{xy}^\ell w_{xy}^\ell}{\sum_{xy} \beta_{xy} (w_{xy}^\ell)^2} ,$$

and

$$\lambda_2 = \min \{-h_q^\ell / d_q^\ell \mid d_q^\ell < 0\} .$$

(C) If the choice of the basic variable is

$$(4.20) \quad \mu_p^\ell = \max \{\mu_q^\ell \mid h_q^\ell > 0, q \in Q_j\} ,$$

instead of (4.15), we will obtain the following direction:

$$(4.21) \quad d_q^\ell = 0 \quad \text{if } q \neq r, q \neq p \quad \text{and}$$

$$q \in \bar{Q}_j$$

$$d_r^\ell = 1$$

$$d_p^\ell = -1$$

where

$$\mu_r = \min \{\mu_q, q \in Q_j\} .$$

In this case the corresponding algorithm would be similar to the one proposed by Dafermos and Sparrow [6] and implemented with a column generation technique by Leventhal, Nemhauser and Trotter [7].

We will now consider the case where the a priori generation and storage of all the possible paths become prohibitive. As mentioned previously, it turns out that it is possible to circumvent this difficulty by the application of a "Restriction" strategy. Basically, this strategy enables paths to be generated economically as needed without having to specify them in advance.

To this end, consider the problem P_j given by (4.13), let $\{h_q^1\}$ be a feasible flow and $Q_j^1 = \{q \mid h_q^1 > 0\}$ be the set of paths with positive flow. Let RP_j be the problem P_j with the additional constraint

$$(4.22) \quad h_q = 0 \quad \text{if } q \notin Q_j^1.$$

Suppose RP_j is solved for an optimal flow $\{h_q^*\}$, then the following Kuhn-Tucker conditions are verified

$$(4.23) \quad \begin{aligned} \mu_q^* &= \sigma_j & \text{if } h_q^* > 0 & \quad q \in Q_j^1 \\ \mu_q^* &> \sigma_j & \text{if } h_q^* = 0 & \quad q \in Q_j^1. \end{aligned}$$

If in addition

$$(4.25) \quad \mu_q > \sigma_j \quad \text{for } q \in Q_j - Q_j^1,$$

is verified, it is clear that $\{h_q^*\}$ is also an optimal flow of P_j .

To verify (4.25), it is sufficient to test

$$(4.26) \quad \mu_r = \min \{\mu_q \mid q \in Q_j\} \geq \sigma_j.$$

If the test is negative, the path r becoming attractive at this stage, has to be added to the current set of available paths to form the new set

$$(4.27) \quad Q_j^2 = Q_j^1 \cup \{r\},$$

and the procedure goes on until optimality is obtained.

Therefore the a priori generation of all the paths, has been replaced here, via (4.26), by a sequence of related shortest path problems. The corresponding algorithm is as follows:

ALGORITHM A3

Step 1 Given a feasible flow $\{h_q\}$, form the sets of available

paths: $Q_j^1 = \{q \mid h_q > 0, q \in Q_j\}$ for all j .

Set $\ell = 1$ and go to step 2.

Step 2 Solve the traffic assignment over the current set of available

paths: $Q^\ell = \bigcup_j Q_j^\ell$ with algorithm A2. At equilibrium,

set $\sigma_j = \min \{\mu_q \mid q \in Q_j^\ell\}$ for all j , and go to step 3.

Step 3 For each O/D pair j , find a shortest path r_j :

$$\mu_{r_j} = \min \{\mu_q \mid q \in Q_j\}.$$

For each O/D pair j , set

$$Q_j^{\ell+1} = \begin{cases} Q_j^\ell \cup \{r_j\} & \text{if } (\mu_{r_j} - \sigma_j) / \sigma > \epsilon \\ Q_j^\ell & \text{otherwise} \end{cases}.$$

If $Q^{\ell+1} = \bigcup_j Q_j^{\ell+1} \subseteq Q^\ell$ terminate, otherwise set $\ell = \ell + 1$ and

return to step 2.

Remarks

(A) Different strategies for the "paths-adding" procedure can be used instead of the one outlined in step 3; for example one may add one path at a time, or instead better, paths originating at the same origin node, since efficient shortest path algorithms usually produce a minimum tree instead of a specific path.

(B) The accuracy of the terminal flow pattern is the same as in algorithm A2 (remark (A)).

4.3 AN ALGORITHM BASED ON THE NODE-ARC FORMULATION

Until now, we have introduced algorithms in which the shortest path finding procedure plays a predominant part. Although, shortest path algorithms are generally effective and easily implemented, if the shortest path finding phase is executed iteratively a large number of times, it becomes time consuming. Fortunately, the shortest path finding issue is not inherent to the traffic assignment problem as it is generally conceived, and the next solution approach will clearly prove it.

The following solution approach has been studied by the author in [14], and much of the development of the next paragraph is borrowed from [14] .

Consider the k^{th} specific single commodity optimization problem associated with the origin s :

$$(4.28) \quad \min \sum_{xy} g_{xy} (f_{xy})$$

subject to

$$(4.29) \quad \sum_{y \in N} f_{xy}^k - \sum_{y \in N} f_{yx}^k = \begin{cases} \sum_t g_{st} & \text{if } x = s \\ -g_{sx} & \text{if } x \in T \\ 0 & \text{otherwise} \end{cases}$$

$$f_{xy}^k > 0$$

$$(4.30) \quad f_{xy} = f_{xy}^k + (\text{constant})$$

where the constant part of (4.30) equals $\sum_{j \neq k} f_{xy}^j$.

It is well known that a basis of the incidence matrix (4.29) corresponds on the network to a spanning tree. Moreover, it is shown in [14], that in the application of the general programming approach using a descent direction obtained by the partition of variables, it is possible to restrict oneself to the set of bases, such that each basis corresponds to a directed spanning tree^{*} rooted at the origin s .

It is also well known that a column $Y_j = (B^{-1}C)_j$ of the Simplex tableau, where B and C are respectively a basis and the remaining columns of the incidence matrix (4.29), correspond on the network to a unique cycle formed by the arc $j = (x, y)$ and the in-tree arcs given by the non zero elements of Y_j . More specifically

$$Y_{ij} = \begin{cases} +1 & \text{if the in-tree arc } i \text{ is oriented in the same} \\ & \text{direction as arc } j \\ -1 & \text{if the in-tree arc } i \text{ is oriented in the} \\ & \text{opposite direction as } j \end{cases}$$

* A directed spanning tree is defined as a spanning tree where all its arcs are oriented along the directed paths connecting the root to all other nodes.

Let define

$$\begin{aligned}
 (4.31) \quad \Gamma_{xy}^+ &= \{ (a,b) \mid (a,b) = i \text{ and } Y_{ij} = -1 \} \cup \{ (x,y) \} \\
 \Gamma_{xy}^- &= \{ (a,b) \mid (a,b) = i \text{ and } Y_{ij} = 1 \} \\
 \Gamma_{xy} &= \Gamma_{xy}^+ \cup \Gamma_{xy}^- .
 \end{aligned}$$

In this case, the computation of the multipliers u_{xy} (expression

(3.12)) becomes

$$(4.32) \quad u_{xy} = \sum_{(a,b) \in \Gamma_{xy}^+} c_{ab} - \sum_{(a,b) \in \Gamma_{xy}^-} c_{ab} .$$

Let $L(x)$ be the path connecting the origin node s to node x , consisting only of in-tree arcs. We define the cost of $L(x)$ as

$$(4.33) \quad \Pi(x) = \sum_{(a,b) \in L(x)} c_{ab} .$$

Obviously (4.32) may be rewritten as

$$(4.34) \quad u_{xy} = c_{xy} + \Pi(x) - \Pi(y) .$$

Therefore, with appropriate labels attached to the nodes of the network, the directed spanning tree will be easily traced back, thus allowing an efficient computation of the relative costs $u_{xy}, (x,y) \in M$, needed to generate the Convex Simplex descent direction (3.13). The major difference between this approach and those previously described, is that the reallocation of flow is not done iteratively over two paths or set of paths but over a loop.

We are now ready to describe the resulting algorithm.

ALGORITHM A5

Step 1 Given a feasible flow $[f_{xy}^j]$ and the set $AR(j)$ of in-tree arcs associated with each commodity j ($j = 1, 2, \dots, ns$), set $i = -1$, $k = 0$ and go to step 2.

Step 2 If $k = ns$, terminate, otherwise set $i = i+1$.
Set $j = i$ modulo $(ns) + 1$ and the iteration counter $\ell = 1$.
Go to step 3.

Step 3 (3.1) Compute $\cdot \Pi(x) = \sum_{(a,b) \in I(x)} c_{ab}$, $x \in N$
 $\cdot u_{xy} = c_{xy} + \Pi(x) - \Pi(y)$
for all $(x,y) \notin AR(j)$
 $\cdot u_{xy} = 0$ $(x,y) \in AR(j)$

(3.2) Define

$$I_1 = \{(a,b) \mid u_{ab} / \Pi(b) < -\epsilon_2\}$$

$$I_2 = \{(a,b) \mid u_{ab} / \Pi(b) > \epsilon_2 \text{ and } f_{ab}^j > \epsilon_1\}$$

where ϵ_1 and ϵ_2 are predetermined positive scalars.

If $I_1 = I_2 = \emptyset$ and $\ell = 1$, set $k = k + 1$ and go to step 2.

If $I_1 = I_2 = \emptyset$ and $\ell > 1$, set $k = 1$ and go to step 2, otherwise go to (3.3).

(3.3) Determine (x_1, y_1) and (x_2, y_2) such that

$$u_{x_1 y_1} = \min \{u_{ab} \mid (a,b) \in I_1\}$$

$$u_{x_2 y_2} f_{x_2 y_2}^j = \max \{ u_{ab} f_{ab}^j \mid (a,b) \in I_2 \} .$$

If $| u_{x_1 y_1} | \geq u_{x_2 y_2} f_{x_2 y_2}^j$, set $(x,y) = (x_1, y_1)$ and $\alpha = +1$,

otherwise set $(x,y) = (x_2, y_2)$ and $\alpha = -1$. Go to (3.4) .

(3.4) Trace back

$$\Gamma_{xy}^+ = \{(x,y)\} \cup L(x) - L(x) \cap L(y)$$

$$\Gamma_{xy}^- = L(y) - L(x) \cap L(y) .$$

Set

$$d_{ab} = \begin{cases} 0 & \text{if } (a,b) \notin \Gamma_{xy}^+ \cup \Gamma_{xy}^- \\ -\alpha & \text{if } (a,b) \in \Gamma_{xy}^- \\ \alpha & \text{if } (a,b) \in \Gamma_{xy}^+ . \end{cases}$$

Go to (3.5) .

(3.5) Determine

$$\sigma = f_{x_3 y_3}^j = \begin{cases} \min \{ f_{ab}^j \mid (a,b) \in \Gamma_{xy}^+ \} & \text{if } \alpha = -1 \\ \min \{ f_{ab}^j \mid (a,b) \in \Gamma_{xy}^- \} & \text{if } \alpha = 1 . \end{cases}$$

If $\sigma > \varepsilon_1$, go to (3.6) .

Determine

$$I_3 = \{ (a, y_3) \mid (a, y_3) \in M , \Pi(a) + c_{a y_3} < \Pi(y_3) \}$$

$$\text{or } f_{a y_3}^j > f_{x_3 y_3}^j \} .$$

(if $I_3 = \emptyset$ replace (x_3, y_3) by its successor on Γ_{xy} and continue in the same manner).

Revise the set of in-tree arcs

$AR(j) = AR(j) \cup \{(x_4, y_4)\} - \{(x_3, y_3)\}$ where (x_4, y_4) is any element of I_3 .

Go to (3.1)

(3.6) Solve the following problem for λ^* and go to (3.7):

$$\text{minimize} \quad \sum_{(a,b) \in \Gamma_{xy}} z_{ab} (f_{ab} + \lambda^* d_{ab})$$

subject to $0 < \lambda \leq \sigma$.

(3.7) For all $(a,b) \in \Gamma_{xy}$:

$$\text{Revise } f_{ab}^j = f_{ab}^j + \lambda^* d_{ab}$$

$$f_{ab} = f_{ab} + \lambda^* d_{ab}$$

$$c_{ab} = c_{ab}(f_{ab}).$$

Set $\ell = \ell + 1$ and go to (3.1).

Remarks

(A) Let z^* be the optimal value of the objective function and z the terminal value given by the above algorithm, it is proved in

[14] that

$$z - z^* < \sum_{j=1}^{ns} (k_1^j \epsilon_1 + k_2^j \epsilon_2)$$

where k_1^j and k_2^j are computed from the flow $\{f_{xy}^j\}$ and the relative costs $\{u_{xy}\}$ at termination.

(B) In the case of linear costs $c_{xy}(f_{xy})$ given by (4.11), the determination of λ^* will be given by

$$\lambda^* = \min \{ \sigma, |\mu_{xy}| / \sum_{(a,b) \in \Gamma_{xy}} \beta_{ab} \} .$$

5- COMPUTATIONAL ASPECTS

5.1 A numerical example

We shall illustrate the computational details of each of the developed algorithms by applying them to the numerical example given in Figure 1. Only several iterations will be shown.

For algorithm A1 the marginal link travelling cost is

$$(5.1) \quad c_{xy}(f_{xy}) = \alpha_{xy} + \beta_{xy} f_{xy} ,$$

while for algorithm A3 and A5 we shall use the following function

[15] :

$$(5.2) \quad c_{xy}(f_{xy}) = \alpha_{xy} \left[1.0 + 0.15 \left(\frac{f_{xy}}{\gamma_{xy}} \right)^4 \right]$$

where

α_{xy} is the free flow travel time

γ_{xy} is a parameter related to the capacity.

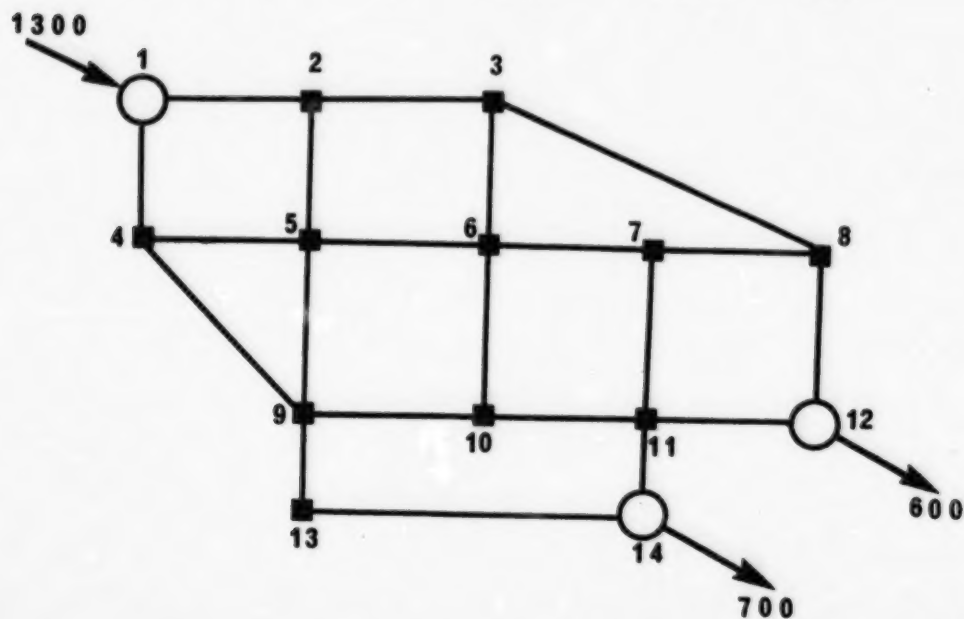
FIGURE 1

NETWORK LAYOUT

Number of Links : 21
 Number of Nodes : 14
 Number of Origins : 1
 Number of Destinations : 2

Origin Nodes are : 1
 Destination Nodes are : 12 14

LINK	FROM	TO	α	γ	β
1	1	2	5.0	300.0	.050
2	1	4	9.0	650.0	.020
3	2	3	9.0	300.0	.040
4	2	5	7.0	650.0	.050
5	3	6	7.0	650.0	.010
6	3	8	14.0	400.0	.040
7	4	5	9.0	300.0	.040
8	4	9	12.0	700.0	.020
9	5	6	3.0	300.0	.030
10	5	9	9.0	650.0	.030
11	6	7	5.0	350.0	.050
12	6	10	13.0	650.0	.020
13	7	8	5.0	350.0	.050
14	7	11	9.0	650.0	.050
15	8	12	9.0	800.0	.020
16	9	10	10.0	400.0	.020
17	9	13	9.0	650.0	.020
18	10	11	6.0	400.0	.010
19	11	12	9.0	400.0	.020
20	11	14	8.0	650.0	.040
21	13	14	11.0	340.0	.040



For all the algorithms, the initial solution is the "all or nothing" assignment illustrated on figure 2.

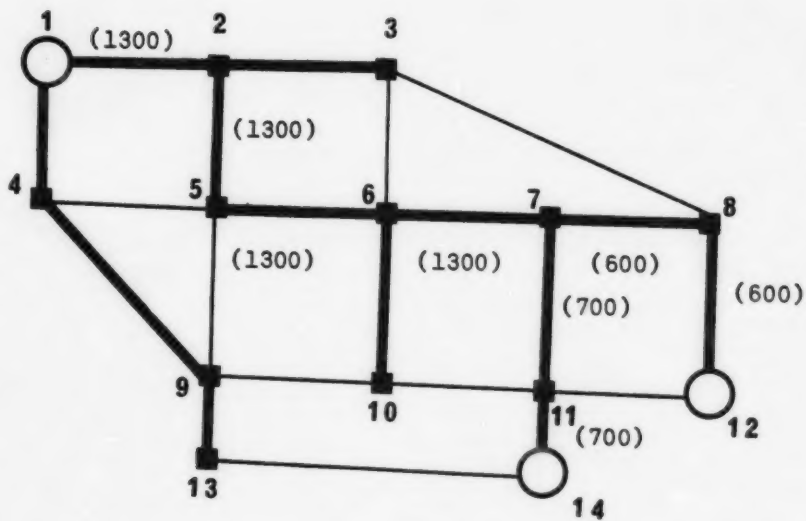


FIGURE 2

The arcs drawn in heavy lines are the shortest path. Numbers in brackets are link flows.

ILLUSTRATION OF ALGORITHM A1

Iteration 1.

Step 1 : compute the new costs c_{xy}^1 :

$$c_{xy}^1 = \alpha_{xy} + \beta_{xy} f_{xy}^1 ,$$

for example $c_{12}^1 = 5 + 0.05 * 1300 = 70.0$.Step 2 : Determine the new shortest paths :

$$P_1 = (1,4) , (4,9) , (9,10) , (10,11) , (11,12)$$

$$P_2 = (1,4) , (4,9) , (9,13) , (13,14) .$$

Compute :

$$w_{xy}^1 = a_{xy}^{P_1} * 600 + a_{xy}^{P_2} * 700$$

for example

$$w_{1,4}^1 = 1 * 600 + 1 * 700 = 1300.0$$

$$w_{9,13}^1 = 0 * 600 + 1 * 700 = 700.0$$

Results are summarized in the following tableau

x	y	$f_{(x,y)}^1$	$c_{(x,y)}^1$	$w_{(x,y)}^1$
1	2	1300.00	70.00	30.00
1	4	0.00	9.00	1300.00
2	3	0.00	9.00	0.00
2	5	1300.00	72.00	0.00
3	6	0.00	7.00	0.00
3	8	0.00	14.00	0.00
4	5	0.00	9.00	0.00
4	9	0.00	12.00	1300.00
5	6	1300.00	42.00	0.00
5	9	0.00	9.00	0.00
6	7	1300.00	70.00	0.00
6	10	0.00	13.00	0.00
7	8	600.00	35.00	0.00
7	11	700.00	44.00	0.00
8	12	600.00	21.00	0.00
9	10	0.00	10.00	600.00
9	13	0.00	9.00	700.00
10	11	0.00	6.00	600.00
11	12	0.00	9.00	600.00
11	14	700.00	36.00	0.00
13	14	700.00	11.00	700.00

Step 3 :
$$\sum_{xy} (w_{xy}^1 - f_{xy}^1) c_{xy}^1 = -3.64 \times 10^5$$

Step 4 :
$$\lambda_1 = - \frac{\sum_{xy} (w_{xy}^1 - f_{xy}^1) c_{xy}^1}{\sum_{xy} (w_{xy}^1 - f_{xy}^1)^2 \beta_{xy}}$$

$$= 0.744$$

$$\lambda^1 = \min \{1, \lambda_1\} = 0.744$$

Step 5 : The new flows are

x	y	$f^{\ell}(x,y)$
1	2	332.65
1	4	967.35
2	3	0.00
2	5	332.65
3	6	0.00
3	8	0.00
4	5	0.00
4	9	967.35
5	6	332.65
5	9	0.00
6	7	332.65
6	10	0.00
7	8	153.53
7	11	179.12
8	12	153.53
9	10	446.47
9	13	520.88
10	11	446.47
11	12	446.47
11	14	179.12
13	14	520.88

The algorithm continues in the same manner.

ILLUSTRATION OF A3

Several iterations have taken place, and a new path with zero flow has just been added to the set Q_1^{ℓ} . We obtain the following situation:

O/D pair number 1 .

$$Q_1^{\ell} = \{q_1, q_2, q_3\} .$$

$$q_1 = (1,2) , (2,3) , (3,8) , (8,12)$$

$$h_{q_1} = 0$$

$$q_2 = (1,4) , (4,9) , (9,10) , (10,11) , (11,12)$$

$$h_{q_2} = 298.8$$

$$q_3 = (1,2) , (2,5) , (5,6) , (6,7) , (7,8) , (8,12)$$

$$h_{q_3} = 301.2$$

In order to obtain a new equilibrium pattern over the new set of paths, algorithm A2 is used.

Iteration ℓ

Step 3 $\mu_{q_1} = 45.0 ; \mu_{q_2} = 52.2 ; \mu_{q_3} = 52.0 .$

$$\mu_{q_1} = \min \{ \mu_{q_i} \} = 45.0$$

$$d_{q_2} = (\mu_{q_2} - \mu_{q_1}) = -7.2$$

$$d_{q_3} = - (\mu_{q_3} - \mu_{q_1}) = -7.0$$

$$d_{q_1} = - (d_{q_2} + d_{q_3}) = 14.2$$

Step 4 Solving

$$\min \sum_{xy} z_{xy} (f_{xy}^{\ell} - \lambda w_{xy}^{\ell})$$

$$\text{subject to } 0 < \lambda \leq \min \left\{ -\frac{h_q}{d_q} \mid d_q < 0 \right\} = 41.6 ,$$

$$\text{one obtains } \lambda^{\ell} = 11.3$$

Step 5 The new flows are

$$h_{q_1} = 160.07$$

$$h_{q_2} = 217.60$$

$$h_{q_3} = 222.32$$

Iteration $\ell + 1$

$$\text{Step 3 } \mu_{q_1} = 51.14 ; \mu_{q_2} = 49.50 ; \mu_{q_3} = 53.19 .$$

$$\mu_{q_2} = \min \{ \mu_{q_i} \} = 49.50 .$$

$$d_{q_1} = 0 \quad \text{because} \quad \frac{(51.14 - 49.50)}{49.50} < 0.05$$

$$d_{q_3} = -3.69$$

$$d_{q_2} = 3.69$$

Step 4 $\lambda^* = \min \{60.28, 5.38\} = 5.38$

Step 5 The new flows are

$$h_{q_1} = 160.07 ; h_{q_2} = 238.77 ; h_{q_3} = 201.16 .$$

Iteration $\ell + 2$

Step 3 $\mu_{q_1} = 49.33 ; \mu_{q_2} = 50.06 ; \mu_{q_3} = 50.47$

$$d_{q_i} = 0 , i = 1, 2, 3$$

thus equilibrium is reached and one pass to the second O/D pair, and so on.

ILLUSTRATION OF ALGORITHM A5

The spanning tree associated with the initial all or nothing solution is shown on figure 2.

Iteration 1

The results of steps (3.1) and (3.2) are shown in the following tableau:

x	y	c_{xy}	$\Pi(x)$	$\Pi(y)$	u_{xy}	f_{xy}^1
3	6	7.00	278.45	454.93	-169.47	0.00
3	8	14.00	278.45	614.15	-321.70	0.00
4	5	9.00	9.00	293.25	-275.25	0.00
5	9	9.00	293.25	21.00	281.25	0.00
9	10	10.00	21.00	467.93	-436.93	0.00
10	11	6.00	467.93	613.49	-139.56	0.00
11	12	9.00	613.49	623.58	-1.09	0.00
13	14	11.00	30.00	623.10	-582.10	0.00

Step 3.3 $u_{x_1 y_1} = \min \{u_{xy}\} = -582.10$

$$(x_1, y_1) = (13, 14)$$

$$u_{x_2 y_2} \cdot f_{x_2 y_2}^1 = \max \{u_{xy} \cdot f_{xy}^1\} = 0$$

Therefore, $(x, y) = (x_1, y_1) = (13, 14)$ and
 $\alpha = 1$.

Step 3.4

$$\Gamma_{xy}^+ = \{ (13, 14), (9, 13), (4, 9), (1, 4) \}$$

$$\Gamma_{xy}^- = \{ (11, 14), (7, 11), (6, 7), (5, 6), (2, 5), (1, 2) \}$$

Step 3.4 $\sigma = f_{11, 14}^1 = \min \{f_{ab} \mid (a, b) \in \Gamma_{xy}^-\} = 700$

Step 3.6 The optimal solution of

$$\min_{(a,b) \in r_{13,14}} \sum_{(a,b) \in r_{13,14}} z_{ab} (f_{ab} + \lambda d_{ab})$$

subject to $0 < \lambda \leq 700$

is $\lambda^* = 664.31$

Step 3.7

The new flows are shown below :

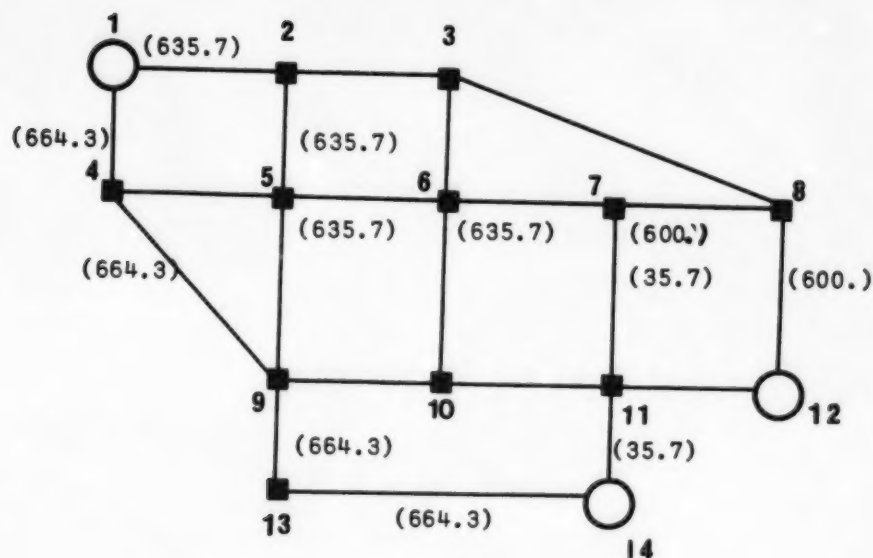


FIGURE 4

This terminates the first iteration, and the algorithm continues in the same manner.

5.2 CODING CONSIDERATIONS AND COMPUTATIONAL RESULTS

The algorithms A1 , A3 , A4 (a version of A3 using the descent direction (4.17)) and A5 were coded in Fortran and tested on the CDC CYBER 74 .

This section is devoted to a discussion of major coding considerations, relevant to the effective implementation of these algorithms and computational results. In particular, the core storage requirements will be considered in some details.

The total core storage required by the codes is classified into

- A- the network structure block
- B- the assignment data block
- C- the cost coefficients block
- D- the working block.

The first three divisions are approximately the same for all the algorithms, while the working memory space is specific to each code.

The network is described by two vectors, where the i^{th} elements contain the indexes of the origin node and the end node of link i . Links are presorted in increasing order of the origin node index. The assignment data includes the set of origins and destinations and the O/D demand matrix. However, the O/D matrix need not be stored, if it is necessary to store the flow of each commodity, either in the form of path flows or flows related to each origin. In this case,

the O/D matrix is only used for the generation of the first solution, it can be entered gradually row by row, processed and later discarded. The area dedicated to the link costs coefficients depends on the cost functions used. Usually travel time functions require at least two parameters: the free flow travel time and a parameter related to the link capacity.

Algorithm A1 is the least sophisticated, among all the developed algorithms, and it requires a relatively small quantity of working memory. From this viewpoint, algorithm A1 can handle rather large networks (few thousands of links and nodes).

In algorithm A3, the main part of the working area in core is allocated to the storage of the links of the generated paths. A simple list structure is used to cope with the unequal number of links on each generated path, and the unequal number of paths related to each O/D pair. Even for a small number of O/D pairs, the area reserved for the working block is quite important and is responsible for the limited capacity of this code.

Algorithm A5 requires a working area smaller than A3 and larger than A1. Its working block is characterized by the memory space reserved for the flow and the spanning tree related to each origin. However, as one origin is processed at a time, the use of auxiliary memory can be easily implemented [14]. In this case the high speed memory required is approximately the same as for algorithm A1.

Algorithm	Memory words required by				Total number of memory words required
	Network structure	Assignment data	cost data	working block	
A1	2 m	$s+d+s*d$	$m*r$	$3(n+m) + 1000$	$m(r+5) + 3n + (s+1)(d+1) + 1000$
A3	2 m	s+d	$m*r$	$3(n+m) + 1000 + 2p+3q+l*q$	$m(r+5) + 3n + q(l+3) + 2p+3q+l*q + 1000$
A5	2 m	s+d	$m*r$	$4n+2m + 1000 + s(n+m)$	$m(4+s+r) + n(4+s) + s+d + 1000$

m = number of links
 n = number of nodes
 s = number of origins
 d = number of destinations
 p = number of O/D pairs
 q = maximum number of paths allowed
 l = expected average number of links on each path
 r = number of link cost coefficients.

TABLE 2

The approximate number of memory words required by each of these codes is presented in Table 2. These bounds may easily be decreased by using packed memory words (one word is designed to contain more than one integer value).

Six test problems were solved with the codes. Results are reported in tables 3, 4 and 5. For all the codes, Dijkstra's shortest path algorithm [16] was used. The travelling cost used is the function given by (5.2). Problems P_1 , P_2 and P_3 are fabricated, while the set P_4 , P_5 and P_6 are based on real network data (Hull, Quebec, Canada). The assignment data is restricted in P_4 and P_5 , that is, not all the O/D pairs are present, in order to have problems of increasing complexity.

An exact comparison of the running times of the algorithms can not be easily obtained, mainly because the stopping criteria are different, thus the terminal solutions are not all identical.

In addition, the step length λ does not have the same unit in all the algorithms, therefore requires a different accuracy and different determination times.

However, from the experimental results, we may reasonably conclude that in general, algorithm A5 is much more efficient than the others, algorithms A4 and A3 have approximately the same rate of convergence which is better than the one of algorithm A1. Algorithm A1 seems to exhibit a "tailing" phenomenon. The difference in running times between A5 and the other is mainly explained by the time spent by the other algorithms in computing repetitively shortest paths.

Problem size: number of				Duration in CPU seconds			
links	origins	O/D pairs	Value of $\bar{g}$ $\times 10^5$	Computation λ (1)	Shortest path finding phase	Total	ϵ
P ₁	23	62	1	7	.35932	.09	.35
P ₂	18	54	17	75	1.55222	.45	2.45
P ₃	18	54	17	75	3.61327	6.31	10.93
P ₄	155	376	17	59	.73744	8.34	46.87
P ₅	155	376	22	100	.88545	14.04	95.65
P ₆	155	376	27	702	16.78107	6.43	54.42
P ₆	155	376	27	702	16.74686	18.60	150.15 ⁽²⁾

(1) λ is determined with an accuracy $\delta = 0.0001$.

(2) Run is forced to terminate.

TABLE 3

Experimental results of algorithm A1.

Problem size: number of					Duration in CPU seconds-		
nodes	links	origins	O/D pairs	Value of $\bar{E} \times 10^5$	Computation $\lambda(1)$	Shortest path finding phase	Total ⁽²⁾
ALGORITHM A3							
P ₁	23	62	1	7	.25866	.32	.91
P ₂	18	54	17	75	1.55231	.13	1.27
P ₃	18	54	17	75	3.61309	.73	4.51
P ₄	155	376	17	59	.74092	3.56	14.87
P ₅	155	376	22	100	.81963	5.53	30.80
ALGORITHM A4							
P ₁	23	62	1	7	.25857	.39	1.03
P ₂	18	54	17	75	1.55231	.13	1.26
P ₃	18	54	17	75	3.61294	.71	4.41
P ₄	155	376	17	59	.73229	4.27	15.97
P ₅	155	376	22	100	.92007	5.53	30.27

(1) the accuracy of λ is 0.5

(2) $\epsilon_1 = 0.75$; $\epsilon_2 = 0.05$

TABLE 4
Experimental results of A3 and A4 .

Problem size: number of				Duration in CPU seconds			Total (2)	
nodes	links	origins	O/D pairs	Value of $\bar{x}$ $\times 10^5$	Computation λ (1)	Shortest path finding phase		
P ₁	23	62	1	7	.25805	.05	.01	.37
P ₂	18	54	17	75	1.55176	.08	.10	.63
P ₃	18	54	17	75	3.81799	.17	.16	.68
P ₄	155	376	17	59	.70004	.39	1.03	10.27
P ₅	155	376	12	100	.88681	.60	1.42	14.51
P ₆	155	376	27	702	15.74780	4.20	1.98	32.67

(1) λ is determined with an accuracy $\delta = 0.01$

(2) $\epsilon_1 = .75$. The terminal solution is such that for any two used paths connecting a specific origin to any specific node in the network have at most 5% of difference in the travelling costs.

TABLE 5
Experimental results of A5

REFERENCES

- [1] Wardrop, J.G., "Some Theoretical Aspects of Road Traffic Research", Proc. Inst. Civil Engineers, Part II, Vol. 1, 1952, pp. 325-378.
- [2] Jorgensen, N.O., "Some Aspect of the Urban Traffic Assignment Problem", Graduate Report, Institute of Transportation and Traffic Engineering, University of California, Berkeley, 1963.
- [3] Jewell, W.S., "Models for Traffic Assignment", Transportation Research, Vol. 1, pp. 31-46.
- [4] Dafermos, S.C., "Network Models for Traffic Assignment", Cornell University Department of Operations Research, Technical Report No. 90, Ithaca, N.Y., 1969.
- [5] Bruynooghe, M., A. Gibert, M. Sakarovitch, "Une Méthode d'Affectation du Trafic", in Fourth Symposium on the Theory of Traffic Flow, Karlsruhe, 1968.
- [6] Dafermos, S.C., T. Sparrow, "The Traffic Assignment Problem for a General Network", Journal of Research, National Bureau of Standards-B, Vol. 73B, No. 2, 1969.
- [7] Leventhal, T., G. Nemhauser and L. Trotter, Jr., "A Column Generation Algorithm for Optimal Traffic Assignment", Transportation Science, Vol. 7, No. 2, 1973.

- [8] Zoutendijk, G., Methods of Feasible Directions , Elsevier, 1960.

- [9] Frank, M. and P. Wolfe, "An Algorithm of Quadratic Programming, Naval Research Logistics Quarterly, Vol 3, pp. 95-110, 1956.

- [10] Wolfe, P., "Methods of Nonlinear Programming", Nonlinear Programming, J. Abadie (editor), John Wiley, New York, pp. 97-131, 1967.

- [11] Zangwill, W.I., Nonlinear Programming: A Unified Approach, Prentice-Hall, Englewood Cliffs, N.J., 1969.

- [12] Luenberger, D.G., Introduction to Linear and Nonlinear Programming, Addison-Wesley, 1973.

- [13] Geoffrion, A.M., Perspectives on Optimization, A.M. Geoffrion (editor), Addison-Wesley, 1972.

- [14] Nguyen, S., "An Algorithm for the Traffic Assignment Problem with Convex Costs", Publication #123, Département d'informatique, Université de Montréal, 1973.

- [15] Bureau of Public Roads, Traffic Assignment Manual, 1964.

- [16] Dijkstra, E., "A note on Two problems in Connexion with Graphs", Numerische Mathematik 1, pp. 269-271, 1959.